Asignatura: Fundamentos de algoritmia
Evaluación continua. Cuestionario.

Grupo A.
Profesor: Isabel Pita.

Nombre del alumno:

1. Tenemos una máquina capaz de realizar 10^8 operaciones por segundo en la que ejecutamos programas con distinto orden de complejidad. Los programas deben ejecutarse en un tiempo lo más cercano posible a un segundo. Relaciona cada orden de complejidad con uno de los tamaños de los datos de entrada dados en la siguiente tabla de forma que todos ellos se ejecuten lo mas cerca posible de un segundo.

Complejidad	Tamaño entrada
$\mathcal{O}(n^2)$	15
$\mathcal{O}(n)$	8
$\mathcal{O}(2^n)$	10.000
$\mathcal{O}(n^n)$	100.000.000
$\mathcal{O}(3^n)$	30

   Respuesta:

   a) Si el algoritmo tiene complejidad $\mathcal{O}(n^2)$, se utiliza una entrada entre 1.000 y 10.000 datos, porque $(10^4)^2 = 10^8$.

   b) Si el algoritmo tiene complejidad $\mathcal{O}(n)$, se utiliza una entrada entre 100.000.000 datos. Aunque normalmente no son aceptables entradas de más de 1.000.000 por problemas de espacio.

   c) Si el algoritmo tiene una complejidad $\mathcal{O}(2^n)$, se utilizan entradas de unos 30 datos, aunque en los algoritmos de vuelta atrás se pueden conseguir entradas de hasta 100 datos con un tiempo aceptable debido a las podas que se realizan.

   d) Si el algoritmo tiene una complejidad $\mathcal{O}(n^n)$, no admite mas de 8 o 10 datos. En algunos problemas de vuelta atrás se puede llegar a las 15.

   e) Si el algoritmo tiene una complejidad $\mathcal{O}(3^n)$ es puede llegar hasta los 15 datos.

2. Dadas las siguientes funciones de coste, indica su orden de complejidad más ajustado (representado por la función más sencilla) y ordena los órdenes de complejidad obtenidos utilizando $\subset$.

   a) $f_1(n) = 6$

   b) $f_2(n) = 4n + \log n^2$

   c) $f_3(n) = n^5 + 3n^3 + 2$

   d) $f_4(n) = 2^n + 3^n$

   e) $f_5(n) = 3n \log 5n + \log n$

   Respuesta:

   a) $f_1(n) = 6 \in \mathcal{O}(1)$.

   b) $f_2(n) = 4n + \log n^2 \in \mathcal{O}(n)$.

   c) $f_3(n) = n^5 + 3n^3 + 2 \in \mathcal{O}(n^5)$.

   d) $f_4(n) = 2^n + 3^n \in \mathcal{O}(3^n)$.

   e) $f_5(n) = 3n \log 5n + \log n \in \mathcal{O}(n \log n)$.

   $\mathcal{O}(1) \subset \mathcal{O}(n) \subset \mathcal{O}(n \log n) \subset \mathcal{O}(n^5) \subset \mathcal{O}(3^n)$.

3. Indica si son ciertas o falsas las siguientes afirmaciones (justifica tu respuesta):

   a) $f(n) = 3n^2 + 5n + 2 \in \mathcal{O}(n \log(n))$

b) $f(n) = 3n + 5n\log(n) + \log(n) \in \mathcal{O}(n\log(n))$

c) $f(n) = \log(10) \in \mathcal{O}(1)$

d) $f(n) = 2^n + n^2 \in \mathcal{O}(n^2)$

e) $f(n) = 2^n + 3^n \in \mathcal{O}(2^n)$

f) $f(n) = n^n + 2^n \in \mathcal{O}(2^n)$

Respuesta:

a) Falso, pertenece al orden $\mathcal{O}(n^2)$.

b) Cierto.

c) Cierto, $f(n) = \log(10)$ es una constante.

d) Falso, la complejidad exponencial es mayor que la cuadrática.

e) Falso, los órdenes de complejidad exponenciales son diferentes.

f) Falso, $f(n) = n^n$ es superior al exponencial.

4. ¿Por qué $\mathcal{O}(\log_{10}(n)) \equiv \mathcal{O}(\log_2(n)) \equiv \mathcal{O}(\ln(n))$?

   Respuesta.

   Porque los logaritmos con distinta base se diferencian en una constante.

$$\log_b n = \frac{\log_a n}{\log_a b}$$

5. Dado el problema de contar el número de cifras de un número, resuelto con el siguiente programa:

```
int x; std::cin >> x; int sol = 0;
while (x != 0) {
   x = x / 10; ++sol;
}
```

   Indica:

   a) Qué complejidad tiene el programa en función del valor de la entrada.

   b) Qué complejidad tiene el programa en función del número de dígitos del valor de entrada.

   Respuesta:

   - $\mathcal{O}(\log(x))$ siendo x el valor de entrada. Cada vuelta del bucle el valor de entrada se divide entre 10, por lo tanto el número de vueltas que da el bucle es $\log x$. El coste de cada vuelta del bucle es constante.
   - $\mathcal{O}(d)$ siendo d el número de dígitos del valor de entrada. En cada vuelta del bucle el número de dígitos del número se reduce en uno, por lo tanto el número de vueltas que da el bucle es d. El coste de cada vuelta del bucle es constante.

6. Indica la complejidad en tiempo más ajustada del siguiente algoritmo y justifícala indicando el número de vueltas que da cada bucle y el coste de cada vuelta.

```
int n; std::cin >> n;
for (int i = 1; i < n-1; ++i)
   for (int j = i-1; j < i+1; ++j)
       x += v[i][j];
}
```

   Respuesta:

   El bucle externo da $n - 2$ vueltas, siendo n el valor de entrada. Para calcular el coste del bucle nos falta comprobar si todas las ejecuciones del cuerpo del bucle tienen el mismo coste y en caso afirmativo cuál es el coste de una ejecución del cuerpo del bucle.

El cuerpo del bucle es un bucle for, que da únicamente 2 vueltas y cada vuelta tiene coste constante por tratarse de una operación aritmética y de un acceso a una matriz. Por lo tanto el coste del bucle interno es $2 * 1 = 2 \in \mathcal{O}(1)$

Por lo tanto el coste aproximado del bucle externo es $n * 1 = n \in \mathcal{O}(n)$, es decir lineal en el valor de n, siendo n el valor leído de la entrada.

7. Indica la complejidad en tiempo más ajustada del siguiente algoritmo,y justifícala indicando el número de vueltas que da el bucle y el coste de esa vuelta.

```
int n; std::cin >> n;
for (int i = 1; i < n; i = i*2)  ++x;
```

Respuesta:

La variable de control del bucle se multiplica por dos en cada vuelta del bucle, por lo que el bucle alcanza el valor n en $\log n$ vueltas. El coste de cada vuelta del bucle es constante porque consiste en incrementar una variable. Por lo tanto el coste del bucle completo pertenece al orden $\mathcal{O}(\log n)$.

8. Dadas dos matrices M_1 y M_2 de dimensiones $m_1 \times m_2$ y $m_2 \times m_3$, indica la complejidad en tiempo más ajustada del siguiente algoritmo y justifícala indicando el número de vueltas que da cada bucle y el coste de cada vuelta.

```
for (int i = 0; i < m1; ++i) {
  for (int j = 0; j < m3; ++j) {
     int suma = 0;
     for (int k = 0; k < m2; ++k)
         suma += m1[i][k] * m2[k][j]
     p[i][j] = suma;
  }
}
```

Respuesta:

El bucle interno (variable de control `k`) da `m2` vueltas y cada vuelta tiene coste constante porque consiste en dos operaciones aritméticas y dos accesos a la matriz, todas ellas operaciones de coste constante. Por lo tanto el coste del bucle intermo es del orden de `m2`.

El bucle interno con variable de control `j` da `m3` vueltas. Cada vuelta tiene coste $\max(c_0, \texttt{m2}, c_1)$, porque se trata de tres instrucciones en secuencia, la primera tiene coste constante por ser una asignación, la segunda es un bucle de coste `m2`, y la tercera es de coste constante por ser una asignación. El coste de cada vuelta es por lo tanto `m2`. Como el bucle da `m3` vueltas, el coste de este bucle es `m2`* `m3`.

El bucle externo da `m1` vueltas. Cada vuelta tiene coste `m2` * `m3`. Por lo tanto el coste completo del algoritmo es `m1` * `m2` * `m3`.